Multi-Objective Reinforcement Learning for Data Collection-Energy Consumption Trade-off in UAV-Aided IoT Networks

Babacar Toure*†, Dimitrios Tsilimantos*, Omid Esrafilian†, Marios Kountouris†
*Advanced Wireless Technology Lab, Paris Research Center, Huawei Technologies France
†Communication Systems Departement, EURECOM, Sophia Antipolis, France
{babacar.toure1, dimitrios.tsilimantos}@huawei.com, {omid.esrafilian, kountour}@eurecom.fr

***Abstract*—Unmanned Aerial Vehicles (UAVs) are increasingly pivotal for data harvesting in the context of IoT networks, leveraging their exceptional mobility and flexibility. Traditional and AI-driven approaches for UAV data collection typically focus on optimizing a single or a fixed combination of objectives, such as data quantity, energy efficiency, or data freshness. However, these methods often fall short in balancing the diverse trade-offs in complex, real-world tasks with conflicting objectives. Our work introduces a novel extension of the Discrete Soft Actor-Critic algorithm to the multi-objective framework. This approach enables UAVs to learn and adapt to diverse trade-offs between data collection and energy consumption. Simulation results demonstrate significant improvements in stability and performance over existing deep multi-objective reinforcement learning algorithms.**

***Index Terms*—Data collection, deep reinforcement learning, IoT, multi-objective optimization, UAV, wireless communications.**

I. Introduction

Internet of Things (IoT) devices are increasingly prevalent across various environments, ranging from urban to rural areas, driving a growing demand for efficient and scalable data collection solutions. In this context, Unmanned Aerial Vehicles (UAVs), as wireless data harvesting agents, have attracted significant attention thanks to their mobility, flexibility, and ease of deployment over large areas without the need for additional ground infrastructure [1]. Indeed, UAVs can be efficiently used in a wide range of data collection scenarios, such as agricultural management, smart cities, and environmental monitoring. Nevertheless, UAV-IoT problems have typically been addressed within a single-objective optimization framework, which is much simpler than the inherently multi-objective nature of real-world scenarios. A comprehensive solution to such data harvesting problems should account for and balance trade-offs among key, often conflicting objectives, such as maximizing data collection, minimizing energy consumption, and preserving data freshness [2].

A. Related Work

In wireless data collection with UAVs, conventional methods often rely on graph-theoretic algorithms for trajectory design, such as A*, Dijkstra, and Voronoi diagrams, which are typically supported by heuristics focused on energy efficiency, network delay, or the quantity of collected data (QCD) [2]. Additionally, both scalar and multi-objective optimization techniques have been applied to address data collection challenges. For instance, a trade-off between Age of Information (AoI) and energy consumption is explored in [3] within a single UAV framework, demonstrating that simultaneously minimizing AoI and energy usage is generally not feasible.

Recently, Deep Reinforcement Learning (DRL) algorithms have emerged as efficient solutions for automatically designing UAV trajectories in data collection tasks. These algorithms handle better high complexity and dimensionality, can learn solely through interactions without requiring prior knowledge of the environment's rules, and provide models capable of making near-optimal decisions in real-time. For example, in [4], the authors train a Deep Q Network (DQN) to jointly optimize flight cruise control and environment sensing data collection of a single UAV, outperforming non-linear heuristic-based methods. In the same line of work, [5] outperformed conventional "non-learning"-based baselines by employing a Twin-Delayed Deep Deterministic policy gradient (TD3) in a realistic throughput maximization task during UAV navigation. Unlike other works, [6] explicitly formulates the data collection problem as a multi-objective optimization task, incorporating data collection, energy harvesting, and UAV energy consumption. However, the proposed solution relies on training a Deep Deterministic Policy Gradient (DDPG) algorithm using a scalarized reward with fixed importance weights assigned to each objective. Thus, the algorithm, once deployed, only produces a single policy, which may be inefficient for other importance weights required for balancing the objectives.

In this work, we study a wireless data collection problem where a UAV flies from a starting point to a final destination while collecting data from sensor devices distributed across an urban environment. We propose a Multi-Objective Reinforcement Learning (MORL) agent, trained only once (no retraining or fine-tuning), which is capable of generating efficient behaviors for various desired trade-offs between data collection and battery conservation - two inherently conflicting objectives. To the best of our knowledge, this approach is novel, as existing scalar DRL methods are limited to discovering a single solution corresponding to a fixed trade-off between the objectives.

B. Contributions

Our main contributions can be summarized as follows:

- We formulate a UAV-based wireless data harvesting problem in an urban area as a multi-objective trajectory optimization problem between data collection and battery consumption objectives.
- We extend the well-known Soft Actor-Critic for discrete action space to support multiple objectives, which, to the best of our knowledge, has not been explored before. We evaluate our proposed MORL algorithm against a carefully designed rule-based method and established MORL baselines, with a focus on performance, stability, and computation needs.
- We further employ target entropy decay and propose exploration improvements through a "heated-up" softmax mechanism in the actor network to significantly boost performance. We conduct a performance study on a UAV wireless data harvesting simulation, demonstrating the superiority of our algorithm compared to baselines.

C. Notations

In the rest of the paper, vectors are represented with bold characters. We denote $[\cdot]^T$ the transpose operator and $\mathbf{1}_l$ the 1-vector of size $l \in \mathbb{N}$, e.g. $\mathbf{1}_3 = [1,1,1]$. The j-th element of a vector $\boldsymbol{v}$ is designated by v_j. For a position vector $\boldsymbol{p}$, p_x, p_y, and p_z denote the x, y, and z coordinates, respectively. For an action-value function Q, the vector of action-values of the different actions in a state s is denoted $\mathbf{Q}(s,\cdot)$. Whenever a function defined on a subset of $\mathbb{R}$ is applied on a vector or matrix, we intend an element-wise application.

II. System model

A. Environment and UAV model

Our scenario closely resembles the one proposed in [7], with the key distinction that we focus on a single UAV and address multiple objectives simultaneously. We consider a data harvesting mission in an urban area engaging one UAV and K static devices at ground level. The k-th device coordinates are denoted $\boldsymbol{p}^k = (x^k, y^k, 0) \in \mathbb{R}^3$. We assume that time is discretized into equal-duration slots of length Δt, indexed by $n \in [0,T]$, with $T \in \mathbb{N}$ being the mission duration. Each device k has a data buffer initially filled with data quantity D_0^k and at time step n, the remaining data quantity is denoted as D_n^k. The UAV is assumed to fly at a constant height h, with its coordinates at time step n given by $\boldsymbol{p}_n = (x_n, y_n, h)$.

The data harvesting mission starts at an initial position $\boldsymbol{p}^I = (x^I, y^I, h)$ with the UAV's battery fully charged at b_0. The mission stops if the UAV's remaining battery, which is denoted as b_n at time step n, is 0 or if the UAV reaches the destination $\boldsymbol{p}^F = (x^F, y^F, h)$. We assume that at each time step n, the UAV can either hover or travel within a distance c in one of the four directions. Thus, we define the set of UAV actions as

$$\mathcal{A} = \left\{ \underbrace{\begin{bmatrix}0\\0\\0\end{bmatrix}}_{\text{hover}}, \underbrace{\begin{bmatrix}0\\c\\0\end{bmatrix}}_{\text{north}}, \underbrace{\begin{bmatrix}-c\\0\\0\end{bmatrix}}_{\text{west}}, \underbrace{\begin{bmatrix}0\\-c\\0\end{bmatrix}}_{\text{south}}, \underbrace{\begin{bmatrix}c\\0\\0\end{bmatrix}}_{\text{east}} \right\}. \tag{1}$$

We further assume that moving consumes a single unit of battery and hovering consumes half a unit. Then, given $\boldsymbol{a}_n \in \mathcal{A}$, the position of the UAV and the remaining battery evolve as:

$$\boldsymbol{p}_{n+1} = \boldsymbol{p}_n + \boldsymbol{a}_n, \tag{2}$$

$$b_{n+1} = \begin{cases} b_n - 0.5 & \text{if } \boldsymbol{a}_n = \text{hover}, \\ b_n - 1 & \text{else.} \end{cases} \tag{3}$$

The consumed battery is denoted by $e_n = b_n - b_{n+1}$.

B. Channel model and data collection

Following [7], the channel gain (in dB) between device k and the UAV at n writes as:

$$g_n^k = \beta_z - \alpha_z \log_{10}(d_n^k) + \eta_z, \tag{4}$$

with $d_n^k = \|\boldsymbol{p}^k - \boldsymbol{p}_n\|_2$ the Euclidean distance between the UAV and device k, index $z \in \{\text{LoS}, \text{NLoS}\}$ denoting whether the communication link is in Line-of-Sight (LoS) or non-LoS (NLoS) condition, α_z the path loss exponent, $\eta_z \sim \mathcal{N}(0, \sigma_z^2)$ the log-normal shadowing and β_z the average gain at a reference distance of 1m. Subsequently, the Signal-to-Noise Ratio (SNR) at time step n is expressed as

$$\text{SNR}_n^k = \frac{P 10^{0.1 g_n^k}}{\sigma^2} \tag{5}$$

where P is the transmit power and σ^2 is the additive white Gaussian noise (AWGN) power at the receiver. The (Shannon) maximum achievable information rate is then

$$R_n^k = \log_2(1 + \text{SNR}_n^k). \tag{6}$$

Adopting the block fading model, we assume that the channel (and hence R_n^k) remains constant during a time slot. At a time step n, the device with the highest SNR is scheduled for transmission. We denote $\delta_n^k \in \{0,1\}$ the scheduling variable, such that $\delta_n^k = 1$ if device k is scheduled and $\delta_n^k = 0$ otherwise. As only one device can be scheduled at a time, it follows:

$$\sum_{k=1}^{K} \delta_n^k \leqslant 1, \quad \forall n \geqslant 0. \tag{7}$$

Note that a device is scheduled for communication only if its SNR at the UAV is greater than a threshold Σ, which effectively defines the communication range. For a scheduled device k, the transmission rate is then

$$Q_n^k = \begin{cases} R_n^k & \text{if } D_n^k \geqslant R_n^k \Delta t \ , \\ D_n^k / \Delta t & \text{else.} \end{cases} \tag{8}$$

C. Problem formulation

We seek the policies π for movement and hovering decisions so as to optimize data collection and battery usage. Using the above notations and definitions, our problem becomes

$$\max_{\pi} \quad \left(\sum_{n=0}^{T-1} \sum_{k=1}^{K} \delta_n^k Q_n^k \Delta t, \quad -\sum_{n=0}^{T-1} e_n \right) \tag{9a}$$

$$\text{s.t.} \quad \boldsymbol{p}_0 = \boldsymbol{p}^I, \boldsymbol{p}_T = \boldsymbol{p}^F, \tag{9b}$$

$$b_T \geqslant 0, \tag{9c}$$

$$(2), (3), (7). \tag{9d}$$

The first objective is to maximize the quantity of collected data during the mission, while the second objective is to minimize the battery consumption. Constraint (9b) enforces that the UAV starts at the initial position and finishes at the final position, (9c) ensures that it reaches the destination before its battery is empty and (9d) encapsulates movement, battery, and data collection rules, respectively. This problem involves multiple layers of complexity, such as the simultaneous optimization of multiple objectives and the uncertainty associated with the channel model. In addition, the two objectives are inherently conflicting, as collecting more data requires higher battery consumption. To address this issue, in the following sections, we propose a novel MORL-based algorithm, which is capable of adapting its policy dynamically to any required trade-off between these objectives without additional training.

III. Multi-Objective Soft Actor-Critic (MOSAC)

A. Multi-Objective Markov Decision Process (MOMDP)

An MOMDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, p, \boldsymbol{R}, \gamma, \mu, f)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p(\cdot|s,a)$ is the transition distribution over next states given a state s and an action a, $\gamma \in [0,1)$ is the discount factor, μ is the initial state distribution, $\boldsymbol{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^M$ is the *multi-objective reward* function (with M the number of objectives) and f is a scalarization/utility function over the objectives. Within this framework, let π denote a policy, which is a function that maps states to actions. The multi-objective action-value function corresponding to this policy can be written for a state $s \in \mathcal{S}$ and an action $a \in \mathcal{A}$ as

$$\boldsymbol{Q}^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i \boldsymbol{R}_{n+i} | s_n = s, a_n = a \right]. \quad (10)$$

It represents the expected discounted return obtained by following the policy π from a given state-action pair. The target solution to such an MODMP is the Pareto set of policies

$$\Pi^* = \{\pi^* | \nexists \pi' : \boldsymbol{V}^{\pi'} \succ_P \boldsymbol{V}^{\pi^*}\}, \quad (11)$$

with $\boldsymbol{V}^\pi = \mathbb{E}_{s_0 \sim \mu}[\boldsymbol{Q}^\pi(s_0, \pi(s_0)]$ the multi-objective value function and $\succ_P$ the Pareto dominance relation:

$$\boldsymbol{V}^{\pi'} \succ_P \boldsymbol{V}^\pi \Leftrightarrow (\forall m : V_m^{\pi'} \geqslant V_m^\pi) \wedge (\exists m : V_m^{\pi'} > V_m^\pi). \quad (12)$$

A policy Pareto-dominates another if it achieves equal or higher values across all objectives while being strictly superior in at least one objective. In general, linear scalarization $f(\boldsymbol{V}^\pi, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{V}^\pi$ is used with $\boldsymbol{w}$ the preference vector such that $\sum_m w_m = 1, w_m \geqslant 0, m = 1, \ldots, M$. Here, w_m is the importance weight of the m-th objective. For a policy π, if no other policy achieves a higher scalarized utility under a given preference $\boldsymbol{w}$, then π is Pareto non-dominated [8].

B. RL formulation

To solve problem (9) using the MOMDP framework, we need to define the states, actions, and rewards of our system. Our MORL agent will learn Pareto optimal policies by observing states, exploring various actions, and learning from the corresponding rewards.

State space. The global state of the system depends on the state of the UAV, the devices, and the communication channels. The state of the UAV at time step n can be described by the remaining battery level b_n, the minimum battery required to reach the destination b_n^{SC}, and the relative distances between the UAV and the destination $d_{n,x} = p_x^F - p_{n,x}$ and $d_{n,y} = p_y^F - p_{n,y}$. Moreover, the device k is described by its relative distances to the UAV $d_{n,x}^k = p_{n,x} - p_x^k$, $d_{n,y}^k = p_{n,y} - p_y^k$, $d_n^k = ||\boldsymbol{p}_n - \boldsymbol{p}^k||_2$, the remaining data in its buffer D_n^k, SNR_n^k and its reachability by the UAV ρ_n^k s.t $\rho_n^k = 1$ if $\text{SNR}_n^k \geqslant \Sigma$ and $\rho_n^k = 0$ otherwise. Thus, we write the state at n as

$$s_n = (s_n^{\text{UAV}}, s_n^u), \quad (13)$$

$$s_n^{\text{UAV}} = (b_n^{SC}, b_n, d_{n,x}, d_{n,y}), \quad (14)$$

$$s_n^u = \{d_{n,x}^k, d_{n,y}^k, d_n^k, D_n^k, \text{SNR}_n^k, \rho_n^k\}_{\forall k}. \quad (15)$$

Note that we assume that the device positions are known by the UAV, but the UAV has no direct access to the current amount of data in the devices' buffers. Instead, the UAV has access to D_0^k, and at $n > 0$, it can infer D_n^k locally based on D_{n-1}^k and the quantity of collected data.

Action space. We employ the action space defined in (1).

Reward. Our vector reward at time step n is an instantaneous version of the objectives defined in (9a):

$$\boldsymbol{r_n} = \left[\sum_{k=1}^{K} \delta_n^k Q_n^k \Delta t, -e_n \right]. \quad (16)$$

Safety controller. As a precaution measure, the UAV is required to execute the sequence of actions that brings it to the destination whenever $b_n^{SC} = 0$, i.e., the remaining battery is just sufficient to reach the destination. However, efficient policies are likely to avoid reaching such a situation altogether.

C. Proposed MOSAC extension

The Soft Actor-Critic (SAC) algorithm [9] is regarded as a benchmark in model-free DRL due to its high sample efficiency and stability. These advantages stem from its unique mixed-learning approach that combines the optimization of expected returns with enhanced exploration, achieved through entropy maximization. Although originally developed for continuous action spaces, SAC has since been adapted to handle discrete action space problems [10]. The mentioned mixed objective is

$$J(\pi) = \sum_{n=0}^{T} \mathbb{E}_{(s_n, a_n) \sim \rho_\pi} [r(s_n, a_n) + \alpha \mathcal{H}(\pi(\cdot|s_n))] \quad (17)$$

where ρ_π is the state-action distribution, $\mathcal{H}(\pi(\cdot|s_n)) = -\sum_a \pi(a|s_n) \log \pi(a|s_n)$ is the entropy of the policy π and α is the trade-off factor between performance (exploitation) and entropy (exploration). The discrete SAC is trained by learning an action-value $Q : \mathcal{S} \to \mathbb{R}^{|A|}$ and a distribution over a discrete action space $\pi : \mathcal{S} \to [0,1]^{|A|}$ with neural networks. Building on the widely adopted method developed in [11], we extend the algorithm to the MORL framework by conditioning both the Q-value and the policy by a preference $\boldsymbol{w}$ so that different policies are learned for different desired weighted combinations of the objectives. As a result, we now learn $Q : \mathcal{S} \times \mathcal{W} \to \mathbb{R}^{M \times |A|}$ and $\pi : \mathcal{S} \times \mathcal{W} \to [0, 1]^{|A|}$.

Soft Q function extension. In the discrete and scalar SAC [10], the Q function is parameterized by θ and learned by gradient descent of

$$J_Q(\theta) = \mathbb{E}_{(s,a,r)\sim\mathcal{B}} \left[\frac{1}{2} \left(Q_\theta(s, a) - y(s, a) \right)^2 \right], \quad (18)$$

with $y(s, a) = r + \gamma \mathbb{E}_{s'} [\boldsymbol{\pi}_\phi(s')^T \left(\mathbf{Q}_{\bar{\theta}}(s', \cdot) - \alpha \log \boldsymbol{\pi}_\phi(s') \right)]$, $\mathcal{B}$ the replay buffer and $\bar{\theta}$ the parameters of the target network [12]. Our extension is written as

$$J_Q(\theta) = \underset{\substack{(s,a,\boldsymbol{r})\sim\mathcal{B} \\ \boldsymbol{w}\sim\mathcal{D}_{\mathcal{W}}}}{\mathbb{E}} \left[\frac{1}{2} \| \boldsymbol{Q}_\theta(s, a, \boldsymbol{w}) - \boldsymbol{y}(s, a, \boldsymbol{w}) \|_2^2 \right], \quad (19)$$

with $\mathcal{D}_{\mathcal{W}}$ a preference distribution and the target given by

$$\boldsymbol{y}(s, a, \boldsymbol{w}) = \boldsymbol{r} + \gamma \mathbb{E}_{s'} \left[\left(\boldsymbol{Q}_{\bar{\theta}}(s', \cdot, \boldsymbol{w}) - \alpha \mathbf{1}_M \log \boldsymbol{\pi}(s', \boldsymbol{w})^T \right) \boldsymbol{\pi}(s', \boldsymbol{w}) \right]. \quad (20)$$

Policy extension. In the scalar SAC, the policy estimator $\pi_\phi(a|s)$ is parameterized by ϕ and trained by minimizing

$$J_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{B}} \left[\boldsymbol{\pi}_\phi(s)^T \left[\alpha \log \boldsymbol{\pi}_\phi(s) - \mathbf{Q}_\theta(s, \cdot) \right] \right]. \quad (21)$$

Such an objective aims to encourage policies with high value and high entropy. We present our extension as

$$J_\pi(\phi) = \underset{\substack{s\sim\mathcal{B} \\ \boldsymbol{w}\sim\mathcal{D}_{\mathcal{W}}}}{\mathbb{E}} \left[\boldsymbol{\pi}_\phi(s, \boldsymbol{w})^T \left[\alpha \log \boldsymbol{\pi}_\phi(s, \boldsymbol{w}) - \boldsymbol{Q}_\theta(s, \cdot, \boldsymbol{w})^T \boldsymbol{w} \right] \right]. \quad (22)$$

Entropy parameter. The trade-off parameter α controls the relative importance of the entropy term. It can either be fixed or learned dynamically with

$$J(\alpha) = \mathbb{E}_{s\sim\mathcal{B}} \left[\boldsymbol{\pi}_\phi(s)^T \left[-\alpha \left(\log \boldsymbol{\pi}_\phi(s) + H_0 \mathbf{1}_{|\mathcal{A}|} \right) \right] \right], \quad (23)$$

where H_0 is a target entropy that controls the desired level of entropy. Taking into account our multi-objective policy:

$$J(\alpha) = \underset{\substack{s\sim\mathcal{B} \\ \boldsymbol{w}\sim\mathcal{D}_{\mathcal{W}}}}{\mathbb{E}} \left[\boldsymbol{\pi}_\phi(s, \boldsymbol{w})^T \left[-\alpha \left(\log \boldsymbol{\pi}_\phi(s, \boldsymbol{w}) + H_0 \mathbf{1}_{|\mathcal{A}|} \right) \right] \right]. \quad (24)$$

D. Heated-up softmax for exploration

In certain scenarios, the discrete SAC may converge to a suboptimal policy due to insufficient exploration or may even be misled into learning an incorrect policy that has a high discounted entropy. Indeed, the objective in SAC can be written as a trade-off between the *discounted* return and the sum of *discounted* entropies $J(\pi) = Q(\pi) + \alpha H(\pi)$, see (17). In the case of sparse, delayed, or small rewards, the algorithm may learn a policy with a high discounted entropy. Surprisingly, in such scenarios, a higher target entropy for more exploration proves ineffective, as the resulting policies already exhibit high discounted entropy relative to the discounted return. Conversely, reducing the target entropy further hampers the exploration.

Algorithm 1: Discrete MOSAC

```
1  Input: preference distribution $\mathcal{D}_w$, target entropies $H_0$
     and $H_{\text{final}}$ , policy temperatures $\tau_0$ and $\tau_{\text{final}}$
2  Initialize replay buffer $\mathcal{B}$, policy temperature $\tau$,
     networks $Q_{\theta_1}$, $Q_{\theta_2}$, $Q_{\bar{\theta}_1}$, $Q_{\bar{\theta}_2}$ and actor network $\pi_\phi$.
3  Match main and target parameters $\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$
4  for episode $e = 1, \dots, M$ do
5      Sample $\boldsymbol{w} \sim \mathcal{D}_w$
6      while not done do
7          Observe state $s$
8          Sample action $a \sim \pi^\tau_\phi(\cdot|s, \boldsymbol{w})$
9          Receive vectorized reward $\boldsymbol{r}$ and observe $s'$
10         Store transition $(s, a, \boldsymbol{r}, s')$ in $\mathcal{B}$
11         Decay policy temperature $\tau$
12         for each gradient step do
13             Sample batch $(s_i, a_i, \boldsymbol{r_i}, s_{i+1}) \sim \mathcal{B}$
14             Sample $J$ preferences $W = \{\boldsymbol{w}_j \sim \mathcal{D}_w\}$
15             Update Q-functions parameters with (19):
                 $\theta_k \leftarrow \theta_k - \lambda_Q \nabla_{\theta_k} J_Q(\theta_k)$ for $k \in \{1, 2\}$
16             Update policy parameters with (22):
                 $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$
17             Learn entropy parameter with (24):
                 $\alpha \leftarrow \alpha - \lambda \nabla_\alpha J_\alpha(\phi)$
18             Update target networks:
                 $\bar{\theta}_k \leftarrow \tau\theta_k + (1 - \tau\bar{\theta}_k)$ for $k \in \{1, 2\}$
```

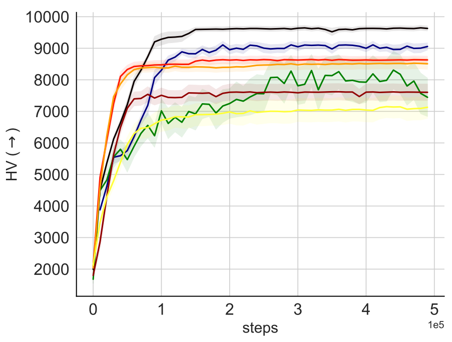
We tackle this phenomenon in two steps. First, as in [13], we decay the target entropy in (24) from an initial value H_0 to a value H_{final} ensuring that, as the training progresses, the trade-off parameter α becomes smaller, thereby placing more focus on the discounted return. Second, to enhance exploration, a heated-up softmax is employed on the policy estimator. In fact, the policy network outputs a vector of logits $l = [l_1, \dots, l_{|\mathcal{A}|}]$, which is then passed through a softmax function to generate the actual stochastic policy: $\pi(a|s) = \frac{\exp l_a}{\sum_i \exp l_i}$. We add exploration with a term τ rendering: $\pi^\tau(a|s) = \frac{\exp(l_a/\tau)}{\sum_i \exp(l_i/\tau)}$. The higher the value of τ, the more uniform the policy becomes. When $\tau = 1$, no additional exploration is added. In practice, training begins with an initial $\tau_0 > 1$, which is linearly decayed to τ_{final} over μ_τ steps.

Algorithm 1 provides the different steps of our approach. Note that as in the scalar SAC, two Q-networks are trained to mitigate overestimation bias (see step 15).

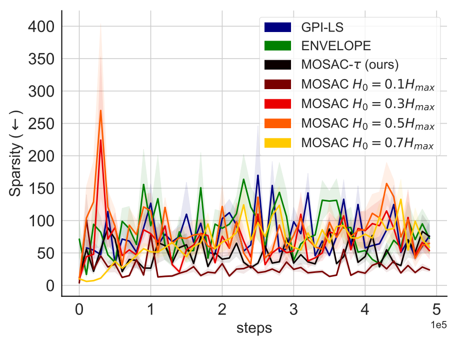
IV. Experiments

A. Baselines and Performance metrics

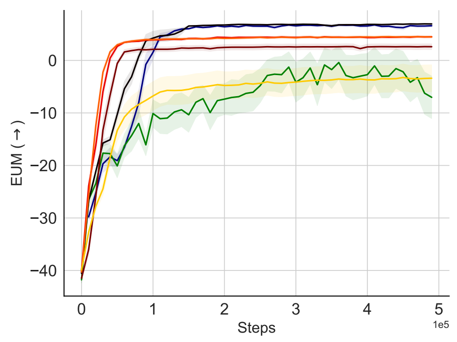
For performance comparison, we employ the following state-of-the-art MORL algorithms for discrete action space: Envelope [14] and GPI-LS [15] using the scripts and parameters from the “MORL-Baselines” package [16]. Our MOSAC algorithm is trained with target entropy levels



(a) Hypervolume



(b) Sparsity



(c) EUM

Fig. 1: Average performance comparison of our proposed algorithm and baseline methods.

TABLE I: Mean and standard deviation of simulation durations

Algo	GPI	ENVELOPE	MOSAC-τ	MOSAC $H_0{=}0.3H_{max}$
Duration (h)	14.4 $\pm$1.23	7.4 $\pm$0.5	4.6 $\pm$0.4	4.53 $\pm$0.1

$\{0.1, 0.3, 0.5, 0.7\}H_{max}$, with $H_{\max} = -\log(1/|\mathcal{A}|)$ the maximum entropy. Its enhanced version from Section III.D is denoted by MOSAC-τ. A fully connected Deep Neural Network (DNN) with 2 layers of 256 neurons each is trained by each algorithm using three different seeds during 500K steps and is evaluated every 10K steps. Our algorithm, akin to the Envelope Algorithm, samples 3 preferences for the DNN update step (see step 14, Alg. 1). The performance is estimated on the set of multi-objective sum of rewards $J^\pi = \{\boldsymbol{J}^\pi_{\boldsymbol{w}}, \boldsymbol{w} \in \mathcal{W}_{test}\}$ with $\mathcal{W}_{test} = \{(\frac{i}{100}, 1 - \frac{i}{100}), i = 0, 1, \dots 100\}$ a coverage of the preference space. Three MORL metrics [17] are employed: i) the Hypervolume $HV(\boldsymbol{J}^{ref}, J^\pi) = \bigcup_{\boldsymbol{J} \in J^\pi} Volume(\boldsymbol{J}^{ref}, \boldsymbol{J})$ indicates the volume of the area covered by the obtained front relative to a given reference point $\boldsymbol{J}^{ref}$, ii) the Sparsity $Sp(J^\pi) = \frac{1}{|J^\pi|-1} \sum_{m=1}^{M} \sum_{i=1}^{|J^\pi|-1} (\tilde{J}^\pi_m(i) - \tilde{J}^\pi_m(i+1))^2$, with $\tilde{J}^\pi_m(i)$ the i-th value in the sorted list for the m-th objective values in J^π, measures the density of the found front and iii) the Expected Utility EUM$(J^\pi) = \mathbb{E}_{\boldsymbol{w} \sim \mathcal{W}_{test}}(\boldsymbol{w}^T \boldsymbol{J}^\pi_{\boldsymbol{w}})$ represents the average scalarized utility. We desire high Hypervolume and EUM with a low Sparsity.

B. Simulations

We consider a scenario with $K = 6$ devices that are randomly scattered over an urban area of size $600m \times 800m$ comprising buildings with various heights [7]. The UAV's initial and destination locations are set as $\boldsymbol{p}^I = (120, 200, 60)m$ and $\boldsymbol{p}^F = (520, 440, 60)m$. The UAV flies at $h = 60m$ and the UAV step size is $c = 20m$. The transmit power is $P = 36dBm$, the SNR threshold is $\Sigma = 15dB$ and the propagation parameters are $\alpha_{\text{LoS}} = 2.5$, $\alpha_{\text{NLoS}} = 3.04$, $\beta_{\text{LoS}} = -30$, $\beta_{\text{NLoS}} = -35$, $\sigma^2_{\text{LoS}} = 2$, $\sigma^2_{\text{NLoS}} = 5$. For MOSAC-$\tau$, we choose $H_0 = 0.6H_{\max}$, $H_{\text{final}} = 0.3H_{\max}$, $\tau_0 = 5$, $\tau_{\text{final}} = 1.5$ and $\mu_\tau = 100$K steps.

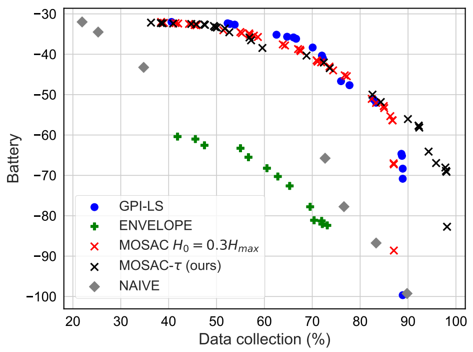
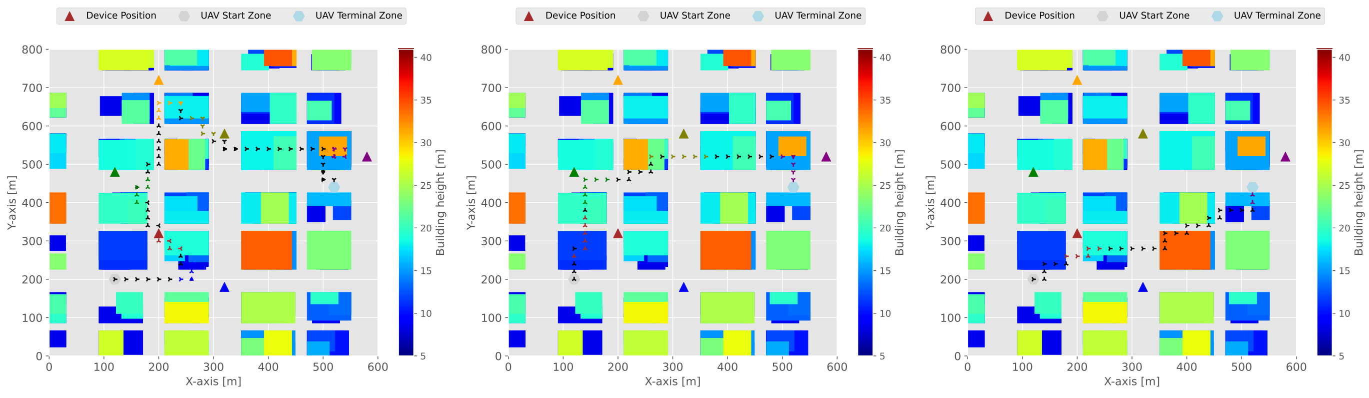


Fig. 2: Average Pareto front comparison over 100 preferences.

1) Multi-objective evaluation: All the algorithms are trained and evaluated by conducting several Monte Carlo simulations, and the average performance is reported in Fig.1. We can conclude that the envelope algorithm has the lowest performance and seems unstable. MOSAC expectedly performs poorly for high and low target entropy levels as hinted in Section III.D and performs best with an entropy level at $0.3H_{max}$. However, it is still outperformed by GPI-LS, supporting the fact that simply tuning the entropy level is not sufficient for performance and exploration in this problem. Also note that after convergence at around 70K steps, MOSAC's performance remains stable indicating that the algorithm does not explore further and becomes trapped in a local minimum. This challenge is successfully tackled by MOSAC-τ outperforming the rest of the algorithms notably on the Hypervolume metric thanks to the enhanced exploration. Indeed, as illustrated on Fig 2, MOSAC-τ manages to find policies that collect close to 100% of the data. However, due to the use of a stochastic policy during evaluation, it achieves a similar EUM performance to GPI. Moreover, Table I shows that, for the same number of training steps (evaluation included), our algorithm runs $2\times$ faster than the Envelope algorithm and $4\times$ faster than GPI. Envelope is penalized by the envelope search [14] over the sampled preferences and GPI-LS by the additional linear support step to update the set of corner weights [15] (preferences). In addition, both algorithms are set



(a) $\boldsymbol{w} = (1, 0)$, $\boldsymbol{J}^{\pi}_{\boldsymbol{w}} = (100\%, -100)$ (b) $\boldsymbol{w} = (0.5, 0.5)$, $\boldsymbol{J}^{\pi}_{\boldsymbol{w}} = (66\%, -41)$ (c) $\boldsymbol{w} = (0, 1)$, $\boldsymbol{J}^{\pi}_{\boldsymbol{w}} = (20\%, -32)$

Fig. 3: MOSAC-τ trajectories with different preferences (initial data quantity $D_0 = 3000$ and battery capacity $b_0 = 100$)

to apply more than one gradient update per training step.

2) Pareto front and trajectory: For each algorithm, we plot the Pareto front averaged across different simulations. We also use a non-learning approach (marked as Naive) that collects from the N-closest devices and then flies to the destination. First, in Fig. 2, we observe in general the superiority of deep MORL approaches compared to the naive non-learning. As already observed, MOSAC-τ dominates other algorithms due to its ability to find policies that collect nearly all the data. In Fig. 3, we plot three trajectories found by our algorithm for different preferences $\boldsymbol{w}$ on one of the random placements of the devices over the city. When $\boldsymbol{w} = (0, 1)$, the UAV takes the shortest path to the destination collecting only from one device on the way and as the inputted preference for data collection increases, the UAV decides to get closer to the devices at the cost of higher battery consumption. For $\boldsymbol{w} = (1, 0)$ all the data is collected consuming all the battery. Interestingly, for $\boldsymbol{w} = (0.8, 0.2)$, our algorithm learns to collect all the data and fly to the destination only consuming 65% of the total battery b_0. Such behaviors could not have emerged from training with a single data collection reward or with pre-determined fixed weights between the objectives.

V. Conclusions

We developed and tested a novel MORL algorithm that combines the performance and stability of the scalar Soft Actor-Critic framework with a heated-up softmax mechanism to enhance exploration efficiency. Our simulation results showed that our approach outperforms MORL baselines in addressing the UAV-assisted wireless data-energy task. Our algorithm efficiently learns policies that capture all desired trade-offs between our objectives in a single training. In future work, we aim to extend this framework to multi-UAV systems, incorporating collaborative strategies for multi-agent, multi-objective learning.

References

[1] J. Luo *et al.*, "Path Planning for UAV Communication Networks: Related Technologies, Solutions, and Opportunities," *ACM Comput. Surv.*, vol. 55, no. 9, Jan. 2023.

[2] K. Messaoudi *et al.*, "A survey of UAV-based data collection: Challenges, solutions and future perspectives," *Journal of Network and Computer Applications*, vol. 216, p. 103670, 2023.

[3] X. Zhang, Z. Chang, T. Hämäläinen, and G. Min, "AoI-Energy Tradeoff for Data Collection in UAV-Assisted Wireless Networks," *IEEE Transactions on Communications*, vol. 72, no. 3, pp. 1849–1861, 2024.

[4] K. Li, W. Ni, E. Tovar, and M. Guizani, "Joint Flight Cruise Control and Data Collection in UAV-Aided Internet of Things: An Onboard Deep Reinforcement Learning Approach," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9787–9799, 2021.

[5] Y. Wang *et al.*, "Trajectory Design for UAV-Based Internet of Things Data Collection: A Deep Reinforcement Learning Approach," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3899–3912, 2022.

[6] Y. Yu *et al.*, "Multi-Objective Optimization for UAV-Assisted Wireless Powered IoT Networks Based on Extended DDPG Algorithm," *IEEE Transactions on Communications*, vol. 69, no. 9, pp. 6361–6374, 2021.

[7] J. Chen *et al.*, "Model-Aided Federated Reinforcement Learning for Multi-UAV Trajectory Planning in IoT Networks," in *2023 IEEE Globecom Workshops (GC Wkshps)*, 2023.

[8] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A Survey of Multi-Objective Sequential Decision-Making," *Journal of Artificial Intelligence Research*, vol. 48, p. 67–113, Oct. 2013.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 1861–1870.

[10] P. Christodoulou, "Soft Actor-Critic for Discrete Action Settings," *ArXiv*, vol. abs/1910.07207, 2019.

[11] A. Abels *et al.*, "Dynamic Weights in Multi-Objective Deep Reinforcement Learning," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[13] Y. Xu *et al.*, "Target Entropy Annealing for Discrete Soft Actor-Critic," in *Deep RL Workshop NeurIPS 2021*, 2021.

[14] R. Yang, X. Sun, and K. Narasimhan, "A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation," in *Neural Information Processing Systems*, 2019.

[15] L. N. Alegre *et al.*, "Sample-Efficient Multi-Objective Learning via Generalized Policy Improvement Prioritization," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023.

[16] F. Felten *et al.*, "A Toolkit for Reliable Benchmarking and Research in Multi-Objective Reinforcement Learning," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2024.

[17] C. F. Hayes *et al.*, "A Practical Guide to Multi-Objective Reinforcement Learning and Planning," *Autonomous Agents and Multi-Agent Systems*, no. 1, 2022.